AMENDMENTS TO THE SPECIFICATION

On page 2, please amend paragraph [0005] as follows:

--It can take a high level of technical skill to create and organize a web site and the pages on that web site. In contrast, adding content can be a simple operation. However, if users of little skill access the web site to add content, they may make a mistake that adversely affects the operation of the web site. Thus, it is desirable to allow technically skilled users to create and organize a web site[[,]] and give them the ability to allow less skilled users to enter content without the possibility of the less skilled users creating problems with the web site.--

On page 4, please amend paragraph [0023] as follows:

--Figure 5 is a flow chart showing the URL modification process.--

On page 4, please amend paragraph [0027] as follows:

--Figure 9 is a block diagram showing an example of how links to other documents [[is]] are automatically generated in a document.--

On page 5, please amend paragraph [0030] as follows:

--In some embodiments, there are multiple levels of users. Higher level users may access all features of the document creation and editing system. These higher level users may create templates, organize the web site, and define what features lower level users have access to. Lower level users have access to fewer than all the features of the system. For example, a lower level user may have the ability to form web pages by entering content related to a single

template[[,]] or a discrete set of templates. With the template, the higher level users have defined what sort of content the lower level user may enter[[,]] and how that content will appear in the web page. The higher level users may prevent the lower level users from making mistakes that adversely affect the operation of the web site and web pages by limiting the access of the lower level users to certain[[,]] narrowly defined functions.--

On page 6, please amend paragraph [0032] as follows:

--Further, documents, such as web pages on the Internet, contain indirect addresses, such as URLs, that refer to other documents and files. It is often useful to modify these addresses. For example, some of these addresses indicate a location relative to other locations. With some such addresses, if the web page moves, the address may not function correctly. Some embodiments of the present invention modify indirect addresses in documents, such as URLs in web pages, as the location of the document changes or under other circumstances so that the indirect addresses continue to function correctly. In the described embodiment, the indirect addresses are URLs, and the documents are web pages on the Internet. However, the invention is not limited to such an application, and also applies to other types of documents with other indirect addresses and on other networks.--

On page 8, please amend paragraph [0044] as follows:

--URL: Stands for "Uniform Resource Locator." The URL defines the location of a file. The URL is a type of indirect address, and refers to the direct[[,]] address. While the term URL is commonly used in the description below, other types of indirect addresses may also be used.--

On page 13, please amend paragraph [0067] as follows:

--When the user creates template files or content files, the files are stored in an organizational structure. The file structure manager module 115 keeps track of the organization of the files, and also keeps track of when files are added, deleted or edited. Organizational structures can take the form of a folder and file system as is well known in the art and used with both Microsoft® Windows® operating systems and Apple® ~~Macintosh~~ Mac OS® operating systems. The user defines the location of each template file, content file, and web page that results from the merging of a template file and content file. Content files can be treated as web pages for organizational purposes, since it is known which template the content files will be merged with to form the web pages. Thus, the content files can be treated as having the same location within the organizational structure as the web pages that will result from the merging of the template with that content file. When the user creates or edits a content file or template file, or merges a template file and content file to create a web page, the template module 118, content module 120, or web page module 114 notifies the file structure manager module 115 of the change. In some situations, when a file is changed, other files should be automatically changed as well. By keeping track of which files are changed, the file structure manager module 115 can notify the template module 118, content module 120 or web page module 114 to change the template file, content file, or web page as needed. The template module 118 or content module 120 can then automatically change the template file or content file in one of the databases 138 and 142, then the web page module 114 can automatically merge the changed template file and/or content file to create the updated web page.--

On pages 15 through 16, please amend paragraph [0073] as follows:

--In some embodiments, there are different user levels. Higher level users have access to more of the system functions than lower level users. For example, higher level users create the templates, while lower level users do not have the ability to create templates. Higher level users also may organize the files of the web site[[,]] and define what functions a lower level user may use.--

On page 22, please amend paragraph [0087] as follows:

--By storing template file 302 separately from content files 304, the template can be modified once and, when the web page module 114 generates the web pages by combining the template and content files, the changes will [[be]] affect every web page based on that template file 302.--

On pages 22 through 23, please amend paragraph [0089] as follows:

--Figure 3(b) is a block diagram of the web pages when viewed, staged or published. To initially form the web page 306, the web page module 114 merges a content file 304 from the content file 120 with a template file 302 from the template module 118 to form the web page 306. In some embodiments, the web page module 114 modifies URLs as the web page module 114 forms the web page 306. The web page module 114 modifies URLs differently based on whether the request came from the preview module 130 or the initiate publish module 132, and also on how the web page 306 will be viewed.--

On pages 26 through 27, please amend paragraph [0098] as follows:

--Figure 4(a) is a screenshot detailing how a user defines a template. In a navigation area 402, the user can access a file organizational structure. This structure can be a file tree, with folder and file organization similar to that found on systems such as popular operating systems ~~used on both~~ Microsoft® Windows® and Apple® ~~Macintosh~~ Mac OS® ~~systems~~. Other file organizational structures can also be used. With the navigation area 402, the user can navigate through the web pages and templates stored in the web publishing system 100, choose to edit existing pages and templates or create new pages and templates, and decide where within the organizational structure to store such templates and pages. A command area 404 provides additional functions available to the user.--

On pages 27 through 28, please amend paragraph [0102] as follows:

--Figure 4(b) is a screenshot showing a template preview presented to the user after the user clicks on the preview button 410 of Figure 4(a). After the user clicks on the preview button 410, the client interface module 124 sends a preview request to the communication module 134, which passes the request to the preview module 130. The preview module 130 then sends the request to the web page module 114. The web page module retrieves the template file from the template module 118. The web page module [[118]] 114 then sends the template file to the communication module 134, which sends the template file to the client interface module 124. The client interface module 124 then displays the template. As the user is in the design UI when previewing a template, the template is displayed nearly the way it would appear as part of a web page generated from the template and a content file. However, instead of being combined with a content file to form the web page, default content 412 is used. The default content is typically

created as part of the template file. The default content takes the place of user-entered content that would normally appear in a web page generated from combining the template and a content file. This allows the user to preview the template without having to also generate a sample content file, and shows where the content from the content file would go in web pages based on the template.--

On pages 30 through 31, please amend paragraph [0108] as follows:

--The user adds content by selecting editable areas on the screen of Figure 4(d). When the user created the template, the user defined [[what]] areas of the template where content will be added to each page. Each of these areas is marked by the page edit symbol 422 in the page editing view. To add content and create a page, or to add information or edit an existing content file, the user selects the page edit symbol 422.--

On pages 31 through 32, please amend paragraph [0110] as follows:

--Figure 4(e) is a screenshot showing how a user enters content to create a content file for a web page after selecting the page edit symbol 422. The content entering screen includes the navigation area 402 and command area 404, as well as a content entering area 424. Information [[427]] on the content entering screen also shows the identity and location within the file structure of the page for which the content is entered. The user may define where in the file structure the page based on the content will be stored, or may define where all pages based on a certain template will be stored. In different embodiments, this can be done when the template associated with the content files is created, when the content file is created, or at other times through use of the file structure manager module 115 to govern the overall file structure. ~~When~~

~~entering content to create a content file for a web page, a user needs little or no knowledge of HTML.~~ A user needs little or no knowledge of HTML to enter content to create a content file for a web page. In the example of Figure 4(e), the user simply types the information into labeled fields. The information entered in this page forms the content file of a web page, as opposed to the template information that is common to each page based on that template. For example, in the field labeled[[,]] "Member_Name" 426, the user types in [[the]] a name. As seen in Figure 4(e), this is a straightforward operation. After typing in all the content, the user clicks on the "Save" button 428. This causes the client interface module 124 to send the content information that the user entered to the communication module 134. The communication module 134 sends the content information, in the form of a content file, to the content update module 128. The content update module 128 then sends the content file to the content module 120, where the content file is stored in the database 142. The user can also delete or edit the content on this screen, and save the revised content. Thus, creation and editing of the web pages is a very simple task.--

On page 35, please amend paragraph [0122] as follows:

--Figure 5 is a flow chart showing a URL modification process 500. When a template and content are combined, the web page module 114 modifies the URLs in the resulting web page as needed. The web page module 114 determines the context 502 in which the web page will appear. There are four contexts: template preview, page preview, local stage or publish, and external stage or publish.--

On page 36, please amend paragraph [0124] as follows:

--When a web page is staged or published, either locally or externally, the web page is not viewed via the user interface. Rather, the web page module 114 forms the web page by combining the template and content, and then sends the web page via the publish module 116 to the staging server 108 or the user server 110. From the staging server 108 or the user server 110, the web page is viewed by viewing clients 112. Thus, under the local and external staging and publishing contexts, the location of the web page depends on the location of the staging server 108 or the user server 110.--

On page 39, please amend paragraph [0136] as follows:

--The URL is checked to determine if the URL is a page relative URL 632. Page relative URLs are identified with a marking code. Preferably, page relative URLs are identified with the marking code of an exclamation point ("!"). If the URL is page relative, the marking code is removed from the front of the URL and the URL is otherwise left untouched 634. This allows the page relative URL to function correctly in its location in a web page. However, the page relative URL will generally not function correctly in a template preview, since the page relative URL references a file in relation to the final web page location, not the template location. If the URL is not a page relative URL, the URL is sent to Figure 6(c) 636.--

On page 42, please amend paragraph [0142] as follows:

--One feature, found in some embodiments, is the automatic linking of documents and the automatic inclusion of content in documents. In one aspect, a document is automatically linked to other documents. ~~Alternately~~ Alternatively, content from other documents is automatically

included in a document. In another aspect, links are automatically created within a single page. ~~Alternately~~ Alternatively, content from within a page is listed or otherwise presented again on that same page. The following description provides a general description of automatic linking. Appendix II provides more detail on how automatic linking is accomplished in one embodiment of the invention.--

On page 42, please amend paragraph [0143] as follows:

--Some embodiments where links to other web pages are automatically generated use the location of the other web pages to accomplish that link generation. Figure 8 is a block diagram showing a simplified overview of an embodiment of a file organizational structure. File organizational structures are well known in the art. In the file organizational structure shown in Figure 8, the general structure is that of folders and files. A first folder 802 (also labeled as "~~folder~~ Folder 1" in Figure 8) is the top level of the organizational structure. The first folder 802 contains documents 804, 806, 808, and 810 and other folders 812, 814, 816, and 818. The folders 812, 814, 816, and 818 may in turn contain documents, other files, and other folders. This can be seen with the fifth folder 818 (also labeled as "Folder 5" in Figure 8). The fifth folder 818 contains documents 822, 824, 826, and 828, as well as a sixth folder 820 (also labeled as "Folder 6" in Figure 8). The sixth folder 820, in turn, can contain still more documents, files and folders.--

On pages 42 through 43, please amend paragraph [0144] as follows:

--The web publishing system 100 organizes the web pages created in a similar organizational structure. Content files can be treated as web pages for organizational purposes,

since it is known which template the content files will be merged with to form the web pages. Thus, the content files can be treated as having the same location within the organizational structure as the web pages that will result from the merging of the template with that content file. Thus, the web pages and content files are treated as being within folders in an organizational structure. The user can see where in the organizational structure content files/web pages are stored by using the navigation area 402, shown in Figures 4(a) through 4(f) ~~and other Figures~~. The navigation area 402 shown in Figure 4(a) shows one top-level folder. The user can click on the top-level folder to see the files and folders within that top-level folder. The user can then click on folders within the top-level folder to see the contents within those folders. The file structure manager module 115 stores the organization of the files used in the web publishing system 100.--

On page 45, please amend paragraph [0150] as follows:

--In addition to specifying the linked documents, the user specifies how the linked documents will be identified in the list document 902. For example, if the linked documents are web pages formed from content files that include a "title" field, the user can specify that the text from the title field be shown in the list document, and that the text shown in the list document is a hypertext link to the linked document. In some embodiments, the list document 902 is created with just a list and no links. Using the above example, the list document 902 then lists the titles of specified web pages, but no links to the specified web pages.--

On page 45, please amend paragraph [0152] as follows:

--In some cases, no ~~other~~ documents may meet the specification for linked documents. The user may define information or content that is to be displayed in such a case. Then, instead of having a blank document when no documents meet the specification, the user-defined information or content is displayed.--

On pages 45 through 46, please amend paragraph [0153] as follows:

--To remain accurate, the list page 902 should be updated when linked documents meeting the specification[[s]] for documents to be listed on the list page 902 are added or changed. In some embodiments, the list page 902 is updated when the user specifies that the list page 902 should be published. When this occurs, the web page module 114 forms the list page 902 from the template and content files. The web page module 114 reads the link specification[[s]] in the template file for the list page 902 and retrieves the relevant files to be linked and their locations from the file structure manager module 115. The web page module 114 then inserts the links to the linked documents into the web page formed from the template and content files.--

On page 46, please amend paragraph [0154] as follows:

--In other embodiments, the file structure manager ~~structure~~ module 115 searches all the templates in the system whenever a content file is created or changed. The file structure manager ~~structure~~ module 115 determines whether the new or changed content should be reflected in any list documents 902. If so, the file structure manager ~~structure~~ module 115 sends instructions to remove the current version of the list document 902 from cache, so that the next

time the list document 902 is needed, it is formed anew with the correct list of links instead of retrieved from cache. The file structure manager ~~structure~~ module 115 may also immediately cause the web page module 114 to form the list document 902 with the correct list of links and send the list document 902 to the publish module 116, which sends the list document 902 on to the user server 110. ~~Alternately~~ Alternatively, the web page module 114 may wait until the created or changed linked documents are published on the user server 110 via the publish module 116, and automatically publish the new, correct list document 902 at the same time.—

On pages 46 through 47, please amend paragraph [0155] as follows:

--Such link pages 902 can also be automatically formed in systems that do not follow the template/content scheme. In such systems, the link page would still include a specification for which pages to link to. A file structure manager ~~structure~~ module 115 can track when pages meeting the specification are saved in the system, and automatically trigger the updating of the link list in the link page. Thus, when a page is created by a user, and the user selects a "save" command, the page is saved. The file structure manager ~~structure~~ module 115 also automatically checks the files in the system to see if the saved page meets the specification within link pages. If so, the link page is automatically updated. This is particularly useful when linking to non-HTML content such as multimedia files (sound files, movie files, etc.) and text documents (documents in .pdf format, in Microsoft® Word® format, etc.).--

On page 48, please amend paragraph [0160] as follows:

--One example of this is a web page with a list of locations for a chain of retail stores. The template would include a field for an address, and specify that the field may be repeated.

When creating the content file, a user would then enter as many addresses as there are locations for the stores. The resultant web page would list each address. Such a field, or multiple fields, that allows multiple instances of content during creation of the content file is called a loop. Each address, in this example, is an instance of the looped fields. Each time the user adds an instance of the looped fields, the user adds a new piece of data for which an entry in the index may be generated.--

On page 48, please amend paragraph [0161] as follows:

--To provide an index in the index document 1002, ~~includes~~ a special tag is included in the template file. This tag specifies that there is a loop in the web page based on the template file. When the web page module 114 forms the web page from the template file and a content file, the web page module 114 creates an index with information from each instance of the loop fields.--